# Table of Contents

Coding Exercise 1: Speed Converter ..... 2
Coding Exercise 2: Megabytes Converter ..... 4
Coding Exercise 3: Barking Dog ..... 6
Coding Exercise 4: Leap Year Calculator ..... 8
Coding Exercise 5: DecimalComparator ..... 9
Coding Exercise 6: Equal Sum Checker ..... 10
Coding Exercise 7: Teen Number Checker ..... 11
Coding Exercise 8: Area Calculator ..... 12
Coding Exercise 9: Minutes To Years and Days Calculator ..... 13
Coding Exercise 10: Equality Printer ..... 14
Coding Exercise 11: Playing Cat ..... 15

# Coding Exercise 1: Speed Converter

1. Write a method called toMilesPerHour that has 1 parameter of type double with the name kilometersPerHour. This method needs to return the rounded value of the calculation of type long.

If the parameter kilometersPerHour is less than 0, the method toMilesPerHour needs to return -1 to indicate an invalid value.

Otherwise, if it is positive, calculate the value of miles per hour, round it and return it. For conversion and rounding, check the notes in the text below.

Examples of input/output:
* toMilesPerHour(1.5); → should return value 1
* toMilesPerHour(10.25); → should return value 6
* toMilesPerHour(-5.6); → should return value -1
* toMilesPerHour(25.42); → should return value 16
* toMilesPerHour(75.114); → should return value 47

2. Write another method called printConversion with 1 parameter of type double with the name kilometersPerHour.

This method should not return anything (void) and it needs to calculate milesPerHour from the kilometersPerHour parameter.
Then it needs to print a message in the format "XX km/h = YY mi/h".
XX represents the original value kilometersPerHour.
YY represents the rounded milesPerHour from the kilometersPerHour parameter.
If the parameter kilometersPerHour is < 0 then print the text "Invalid Value".
Examples of input/output:
* printConversion(1.5); → should print the following text (into the console - System.out): 1.5 km/h = 1 mi/h
* printConversion(10.25); → should print the following text (into the console - System.out): 10.25 km/h = 6 mi/h
* printConversion(-5.6); → should print the following text (into the console - System.out): Invalid Value
* printConversion(25.42); → should print the following text (into the console - System.out): 25.42 km/h = 16 mi/h

* printConversion(75.114); → should print the following text (into the console - System.out): 75.114 km/h = 47 mi/h
Use method Math.round to round the number of calculated miles per hour(double). The method round returns long.

How to use the method round and how it works?

The Math.round() is a built-in math method which returns the closest long to the argument. The result is rounded to an integer by adding 1/2, taking the floor of the result after adding 1/2, and typecasting the result to type long. The method returns the value of the argument rounded to the nearest int value.

USAGE EXAMPLE:

```
double number = 1.5;
long rounded = Math.round(number);
System.out.println("rounded= " + rounded);
System.out.println("with 3.9= " + Math.round(3.9));
System.out.println("with 4.5= " + Math.round(4.5));
int sum = 45;
int count = 10;
// typecasting so result is double e.g. double / int -> double
double average = (double) sum / count;
long roundedAverage = Math.round(average);
System.out.println("average= " + average);
System.out.println("roundedAverage= " + roundedAverage);
```

OUTPUT:

```
rounded= 2
with 3.9= 4
with 4.5= 5
average= 4.5
roundedAverage= 5
```

TIP: In the method printConversion, call the method toMilesPerHour instead of duplicating the code.

NOTE: All methods should be defined as public static like we have been doing so far in the course.

NOTE: 1 mile per hour is 1.609 kilometers per hour
NOTE: Do not add a main method to the solution code.

# Coding Exercise 2: Megabytes Converter

Write a method called printMegaBytesAndKiloBytes that has 1 parameter of type int with the name kiloBytes.

The method should not return anything (void) and it needs to calculate the megabytes and remaining kilobytes from the kilobytes parameter.
Then it needs to print a message in the format "XX KB = YY MB and ZZ KB". XX represents the original value kiloBytes.
YY represents the calculated megabytes.
ZZ represents the calculated remaining kilobytes.

For example, when the parameter kiloBytes is 2500 it needs to print "2500 KB = 2 MB and 452 KB"

If the parameter kiloBytes is less than 0 then print the text "Invalid Value".

EXAMPLE INPUT/OUTPUT

printMegaBytesAndKiloBytes(2500); → should print the following text: "2500 KB = 2
MB and 452 KB"

printMegaBytesAndKiloBytes(-1024); → should print the following text: "Invalid
Value" because parameter is less than 0.

printMegaBytesAndKiloBytes(5000); → should print the following text: "5000 KB = 4
MB and 904 KB"

TIP: Be extremely careful about spaces in the printed message. TIP: Use the remainder operator
TIP: 1 MB = 1024 KB

NOTE: Do not set kilobytes parameter value inside your method. NOTE: The solution will not be accepted if there are extra spaces.

NOTE: The printMegaBytesAndKiloBytes method needs to be defined as public static like we have been doing so far in the course.NOTE: Do not add a main method to solution code.

# Coding Exercise 3: Barking Dog

We have a dog that likes to bark. We need to wake up if the dog is barking at night!

Write a method shouldWakeUp that has 2 parameters.

1st parameter should be of type boolean and be named barking it represents if our dog is currently barking.
2nd parameter represents the hour of the day and is of type int with the name hourOfDay and has a valid range of 0-23.

We have to wake up if the dog is barking before 8 or after 22 hours so in that case return true.

In all other cases return false.

If the hourOfDay parameter is less than 0 or greater than 23 return false.

Examples of input/output:

shouldWakeUp (true, 1); → should return true

shouldWakeUp (false, 2); → should return false since the dog is
not barking.

shouldWakeUp (true, 8); → should return false, since it's not before 8.

shouldWakeUp (true, -1); → should return false since the hourOfDay parameter needs to be in a range 0-23.

TIP: Use the if else statement with multiple conditions.

NOTE: The shouldWakeUp method needs to be defined as public static like we have been doing so far in the course.

NOTE: Do not add a main method to solution code.

# Coding Exercise 4: Leap Year Calculator

Write a method isLeapYear with a parameter of type int named year.

The parameter needs to be greater than or equal to 1 and less than or equal to 9999.
If the parameter is not in that range return false.
Otherwise, if it is in the valid range, calculate if the year is a leap year and return true if it is a leap year, otherwise return false.

To determine whether a year is a leap year, follow these steps:
1. If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
2. If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
3. If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
4. The year is a leap year (it has 366 days). The method isLeapYear needs to return true.
5. The year is not a leap year (it has 365 days). The method isLeapYear needs to return false.

The following years are not leap years:
1700, 1800, 1900, 2100, 2200, 2300, 2500, 2600
This is because they are evenly divisible by 100 but not by 400.

The following years are leap years:
1600, 2000, 2400
This is because they are evenly divisible by both 100 and 400.
Examples of input/output:
* isLeapYear(-1600); → should return false since the parameter is not in range (1-9999)

* isLeapYear(1600); → should return true since 1600 is a leap year

* isLeapYear(2017); → should return false since 2017 is not a leap year

* isLeapYear(2000); → should return true because 2000 is a leap year

NOTE: The method isLeapYear needs to be defined as public static like we have been doing so far in the course.
NOTE: Do not add a main method to solution code.

# Coding Exercise 5: DecimalComparator

Write a method areEqualByThreeDecimalPlaces with two parameters of type double.

The method should return boolean and it needs to return true if two double numbers are the same up to three decimal places. Otherwise, return false.

EXAMPLES OF INPUT/OUTPUT:

* areEqualByThreeDecimalPlaces(-3.1756, -3.175); → should return true since numbers are equal up to 3 decimal places.

* areEqualByThreeDecimalPlaces(3.175, 3.176); → should return false since numbers are not equal up to 3 decimal places

* areEqualByThreeDecimalPlaces(3.0, 3.0); → should return true since numbers are equal up to 3 decimal places.

* areEqualByThreeDecimalPlaces(-3.123, 3.123); → should return false since numbers are not equal up to 3 decimal places.

TIP: Use paper and pencil.
TIP: Use casting.
NOTE: The areEqualByThreeDecimalPlaces method needs to be defined as public static like we have been doing so far in the course.
NOTE: Do not add a main method to solution code.

# Coding Exercise 6: Equal Sum Checker

Write a method hasEqualSum with 3 parameters of type int.

The method should return boolean and it needs to return true if the sum of the first and second parameter is equal to the third parameter. Otherwise, return false.

EXAMPLES OF INPUT/OUTPUT:

* hasEqualSum(1, 1, 1); should return false since 1 + 1 is not equal to 1

* hasEqualSum(1, 1, 2); should return true since 1 + 1 is equal to 2

* hasEqualSum(1, -1, 0); should return true since 1 + (-1) is 1 - 1 and is equal to 0

NOTE: The hasEqualSum method needs to be defined as public static like we have been doing so far in the course.
NOTE: Do not add a main method to solution code.

# Coding Exercise 7: Teen Number Checker

We'll say that a number is "teen" if it is in the range 13 -19 (inclusive).
Write a method named hasTeen with 3 parameters of type int.
The method should return boolean and it needs to return true if one of the parameters is in range 13(inclusive) - 19 (inclusive). Otherwise return false.

EXAMPLES OF INPUT/OUTPUT:

* hasTeen(9, 99, 19); should return true since 19 is in range 13 - 19

* hasTeen(23, 15, 42); should return true since 15 is in range 13 - 19

* hasTeen(22, 23, 34); should return false since numbers 22, 23, 34 are not in range 13-19

Write another method named isTeen with 1 parameter of type int.

The method should return boolean and it needs to return true if the parameter is in range 13(inclusive) - 19 (inclusive). Otherwise return false.

EXAMPLES OF INPUT/OUTPUT:

* isTeen(9); should return false since 9 is in not range 13 - 19

* isTeen(13); should return true since 13 is in range 13 - 19

NOTE: All methods need to be defined as public static like we have been doing so far in the course.
NOTE: Do not add a main method to solution code.

# Coding Exercise 8: Area Calculator

* Write a method named area with one double parameter named radius.
*
* The method needs to return a double value that represents the area of a circle.
*
* If the parameter radius is negative then return -1.0 to represent an invalid value.
*
* Write another overloaded method with 2 parameters x and y (both doubles), where x and y represent the sides of a rectangle.
*
* The method needs to return an area of a rectangle.
*
* If either or both parameters is/are a negative return -1.0 to indicate an invalid value.
*
* For formulas and PI value please check the tips below.
*
* Examples of input/output:
*
* * area(5.0); should return 78.53975
*
* * area(-1);  should return -1 since the parameter is negative
*
* * area(5.0, 4.0); should return 20.0 (5 * 4 = 20)
*
* * area(-1.0, 4.0);  should return -1 since first the parameter is negative
*
*
*
* TIP: The formula for calculating the area of a rectangle is x * y.
*
* TIP: The formula for calculating a circle area is radius * radius * PI.
*
* TIP: For PI use a constant from Math class e.g. Math.PI
*
* NOTE: All methods need to be defined as public static like we have been doing so far in the course.
*
* NOTE: Do not add a main method to your solution code!
*

# Coding Exercise 9: Minutes To Years and Days Calculator

* Write a method printYearsAndDays with parameter of type long named minutes.

* The method should not return anything (void) and it needs to calculate the years and days from the minutes parameter.

* If the parameter is less than 0, print text "Invalid Value".

* Otherwise, if the parameter is valid then it needs to print a message in the format "XX min = YY y and ZZ d".

* XX represents the original value minutes.
* YY represents the calculated years.
* ZZ represents the calculated days.

* EXAMPLES OF INPUT/OUTPUT:

* printYearsAndDays(525600); → should print "525600 min = 1 y and 0 d"

* printYearsAndDays(1051200); → should print "1051200 min = 2 y and 0 d"

* printYearsAndDays(561600); → should print "561600 min = 1 y and 25 d"

* TIPS:

* Be extra careful about spaces in the printed message.

* Use the remainder operator

* 1 hour = 60 minutes

* 1 day = 24 hours

* 1 year = 365 days

* NOTES

* The printYearsAndDays method needs to be defined as public static like we have been doing so far in the course.

* Do not add main method to solution code.

* The solution will not be accepted if there are extra spaces

# Coding Exercise 10: Equality Printer

* Write a method printEqual with 3 parameters of type int. The method should not return anything (void).

* If one of the parameters is less than 0, print text "Invalid Value".

* If all numbers are equal print text "All numbers are equal"

* If all numbers are different print text "All numbers are different".

* Otherwise, print "Neither all are equal or different".

* EXAMPLES OF INPUT/OUTPUT:

* printEqual(1, 1, 1); should print text All numbers are equal

* printEqual(1, 1, 2); should print text Neither all are equal or different

* printEqual(-1, -1, -1); should print text Invalid Value

* printEqual(1, 2, 3); should print text All numbers are different

* TIP: Be extremely careful about spaces in the printed message.

* NOTES

* The solution will not be accepted if there are extra spaces.

* The method printEqual needs to be defined as public static like we have been doing so far in the course.

* Do not add main method to solution code.

# Coding Exercise 11: Playing Cat

* The cats spend most of the day playing. In particular, they play if the temperature is between 25 and 35 (inclusive). Unless it is summer, then the upper limit is 45 (inclusive) instead of 35.

* Write a method isCatPlaying that has 2 parameters. Method needs to return true if the cat is playing, otherwise return false
* 1st parameter should be of type boolean and be named summer it represents if it is summer.
* 2nd parameter represents the temperature and is of type int with the name temperature.

* EXAMPLES OF INPUT/OUTPUT:

* isCatPlaying(true, 10); should return false since temperature is not in range 25 - 45

* isCatPlaying(false, 36); should return false since temperature is not in range 25 - 35 (summer parameter is false)

* isCatPlaying(false, 35); should return true since temperature is in range 25 - 35

* NOTES

* The isCatPlaying method needs to be defined as public static like we have been do